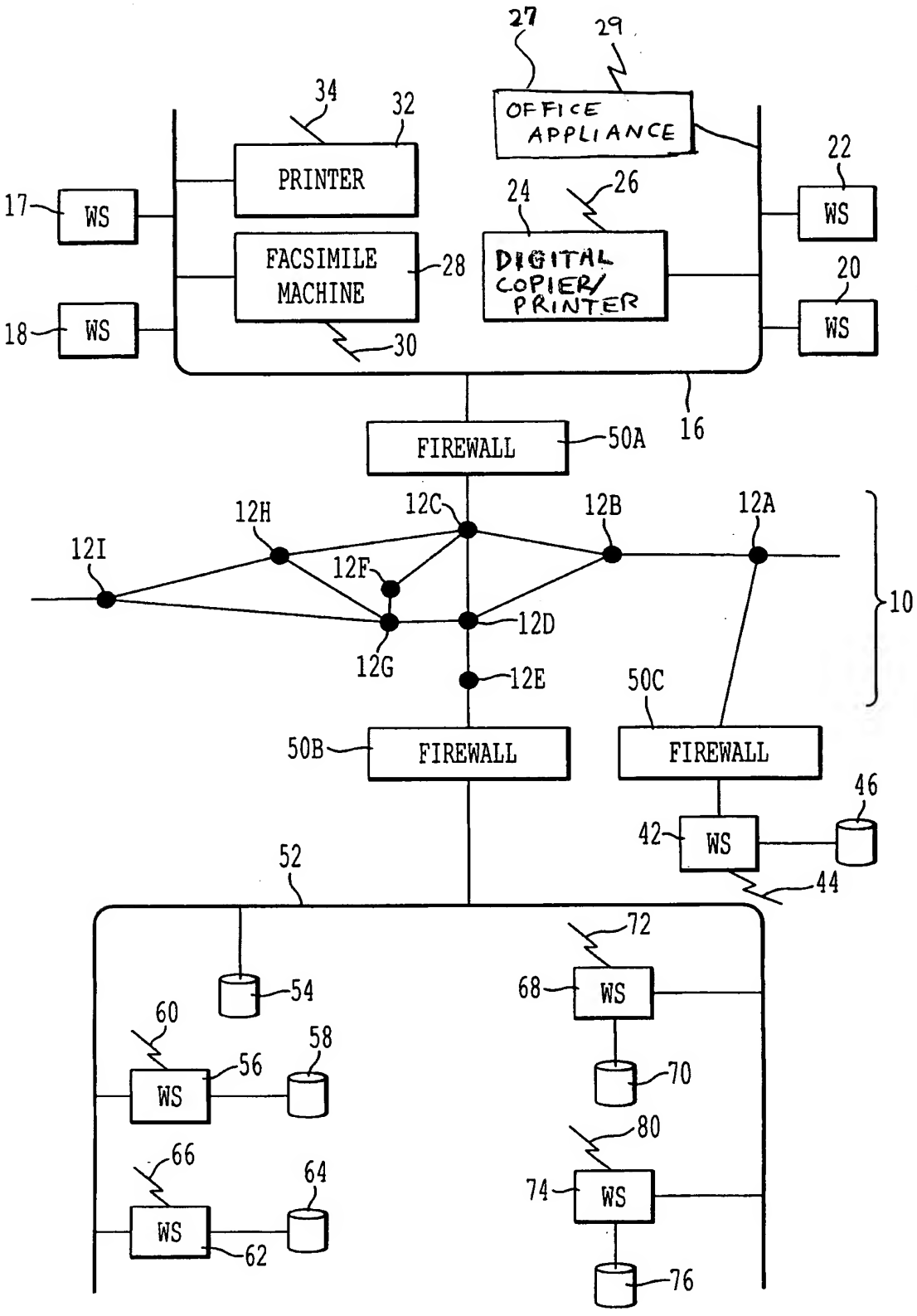
27
29
OFFICE APPLIANCE
34
32
PRINTER
22
17
WS
24
26
DIGITAL COPIER/ PRINTER
WS
FACSIMILE MACHINE
28
20
18
WS
30
WS
16
FIREWALL
50A
12H
12C
12B
12A
12I
12F
10
12D
12G
12E
50C
50B
FIREWALL
FIREWALL
46
42
WS
52
44
72
54
68
WS
60
58
56
WS
70
80
66
64
74
WS
WS
62
76

FIG. 1

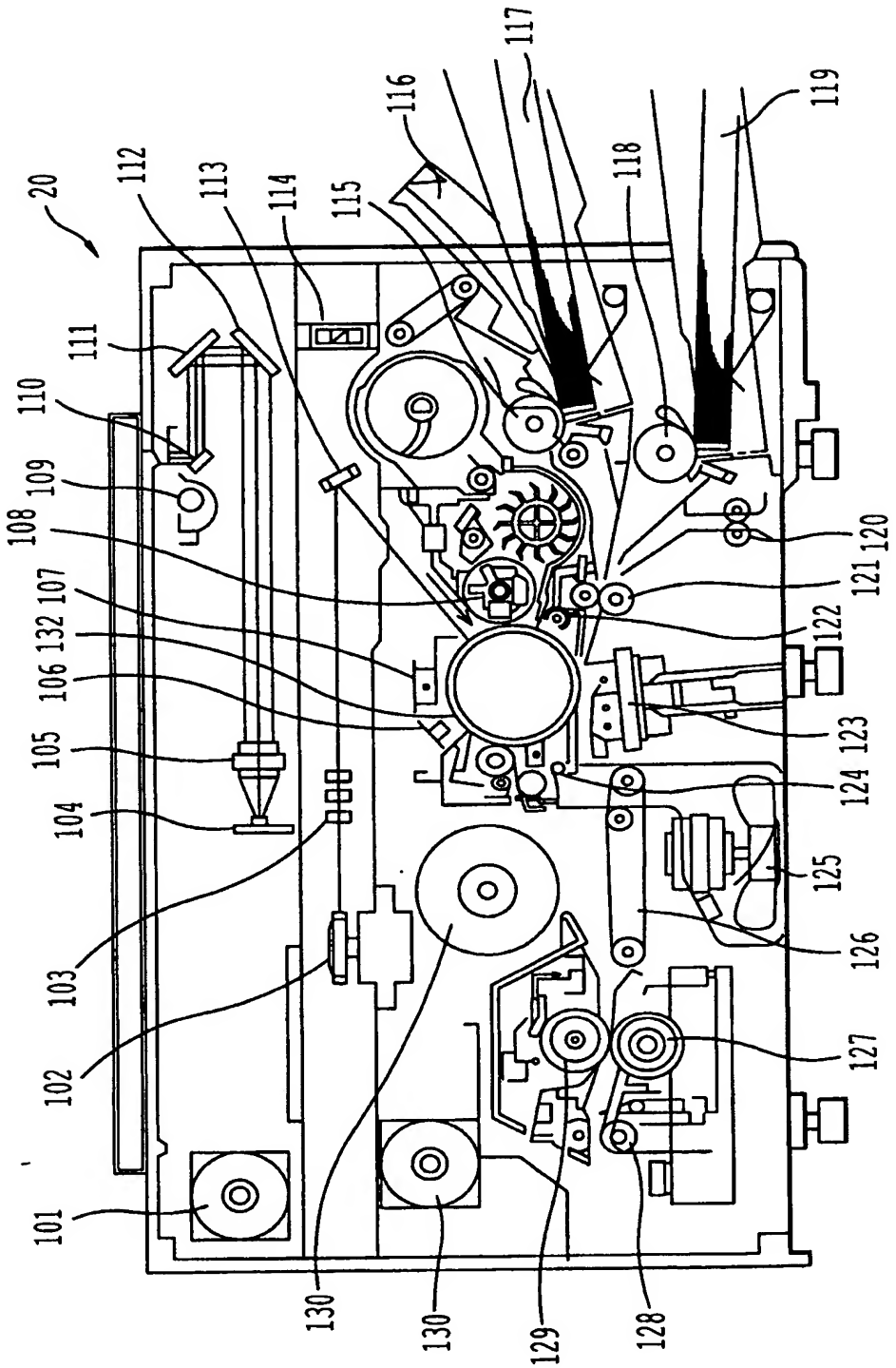
20
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
130
132

FIG. 2

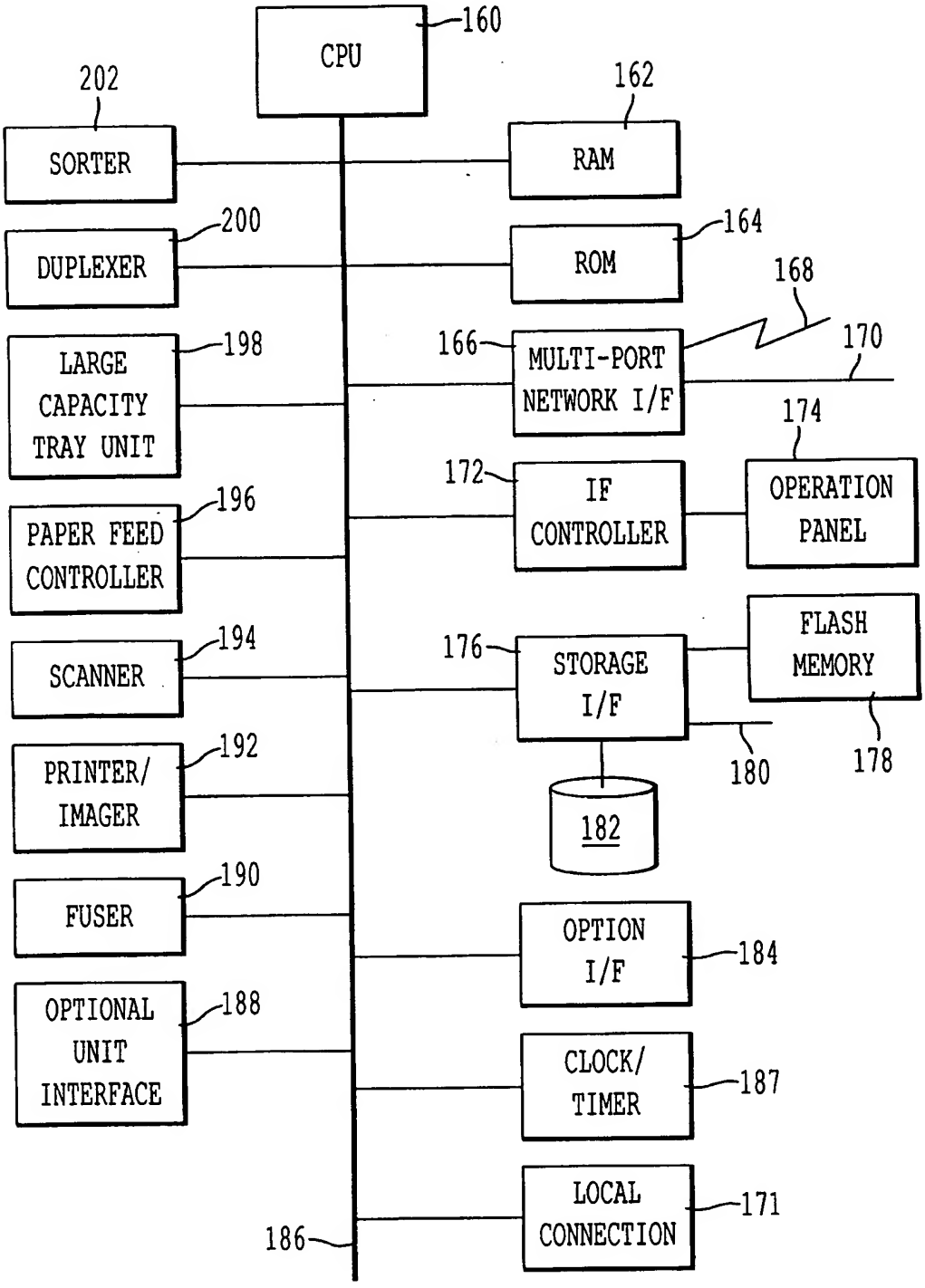
CPU
160
202
162
SORTER
RAM
200
164
DUPLEXER
ROM
168
198
166
170
LARGE CAPACITY TRAY UNIT
MULTI-PORT NETWORK I/F
174
172
196
IF CONTROLLER
OPERATION PANEL
PAPER FEED CONTROLLER
194
176
FLASH MEMORY
SCANNER
STORAGE I/F
192
180
178
PRINTER/ IMAGER
182
190
FUSER
OPTION I/F
184
188
OPTIONAL UNIT INTERFACE
CLOCK/ TIMER
187
LOCAL CONNECTION
171
186

FIG. 3

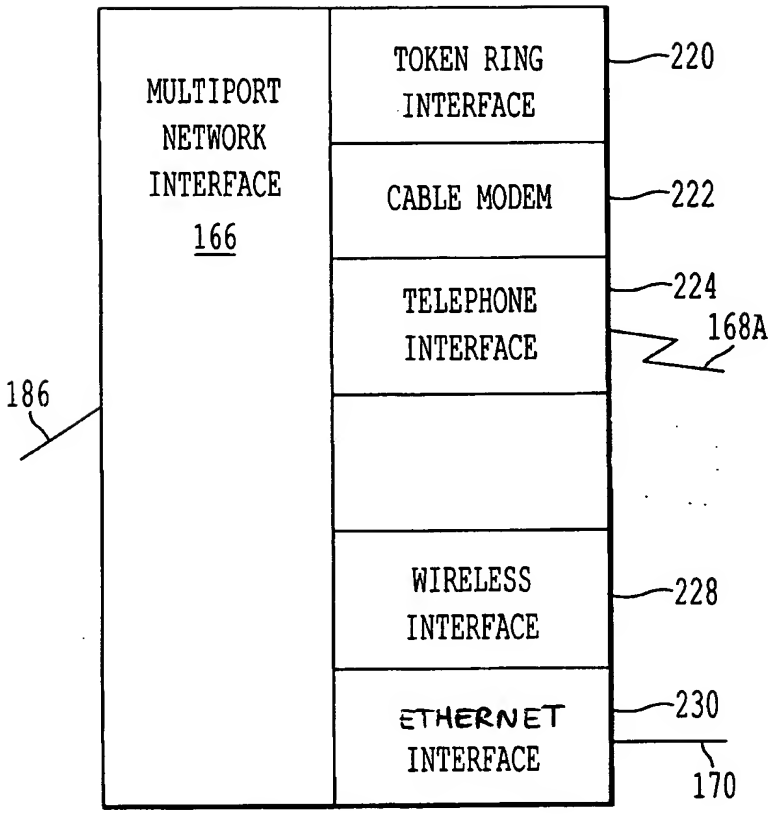
MULTIPORT NETWORK INTERFACE 166
TOKEN RING INTERFACE
220
CABLE MODEM
222
TELEPHONE INTERFACE
224
168A
186
WIRELESS INTERFACE
228
ETHERNET INTERFACE
230
170

FIG. 4

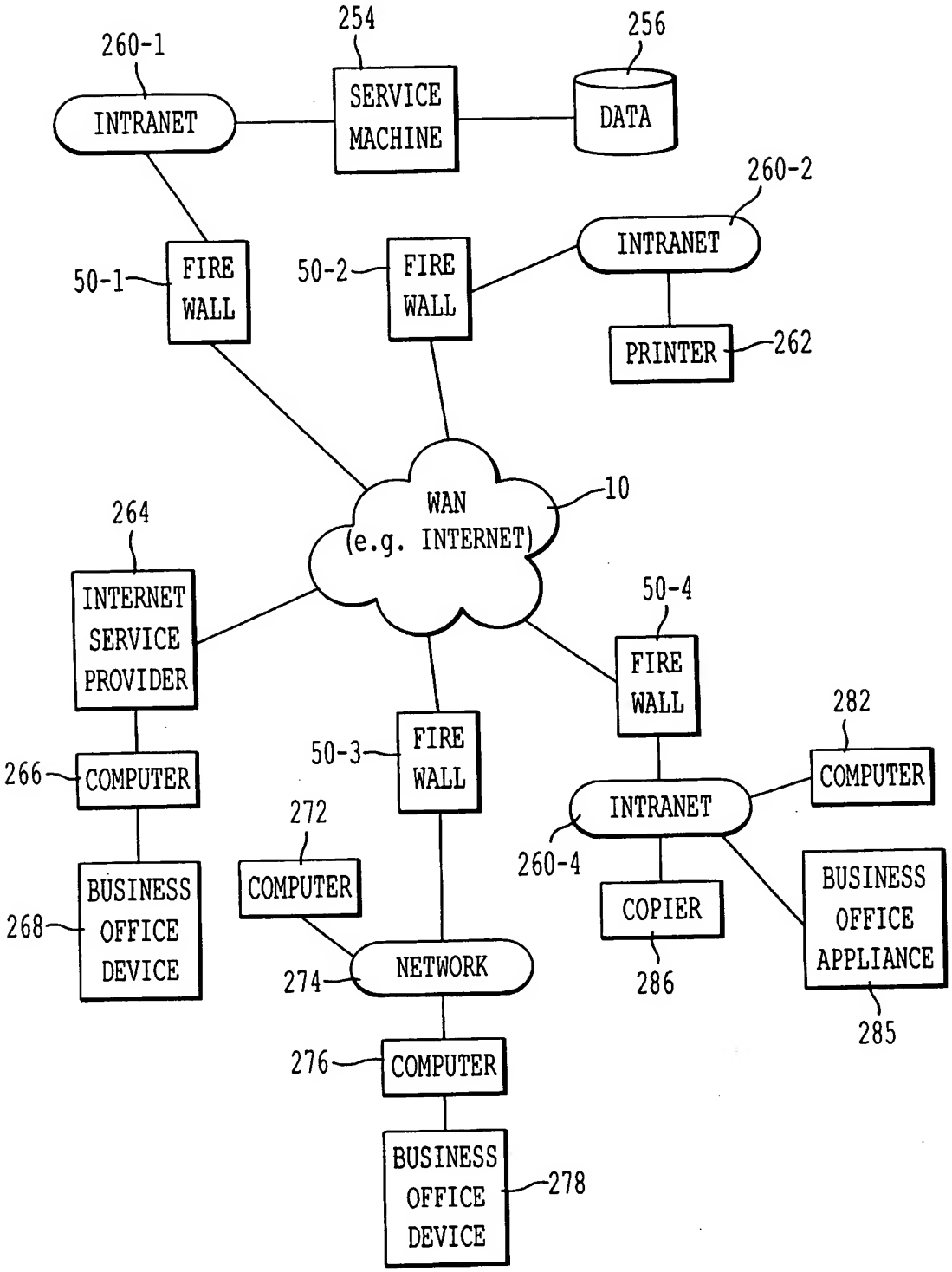
260-1
INTRANET
254
SERVICE MACHINE
256
DATA
260-2
INTRANET
50-1
FIRE WALL
50-2
FIRE WALL
PRINTER
262
WAN (e.g. INTERNET)
10
264
INTERNET SERVICE PROVIDER
50-4
FIRE WALL
282
COMPUTER
266
COMPUTER
50-3
FIRE WALL
INTRANET
260-4
272
COMPUTER
BUSINESS OFFICE DEVICE
268
COPIER
286
BUSINESS OFFICE APPLIANCE
285
274
NETWORK
276
COMPUTER
BUSINESS OFFICE DEVICE
278

FIG. 5

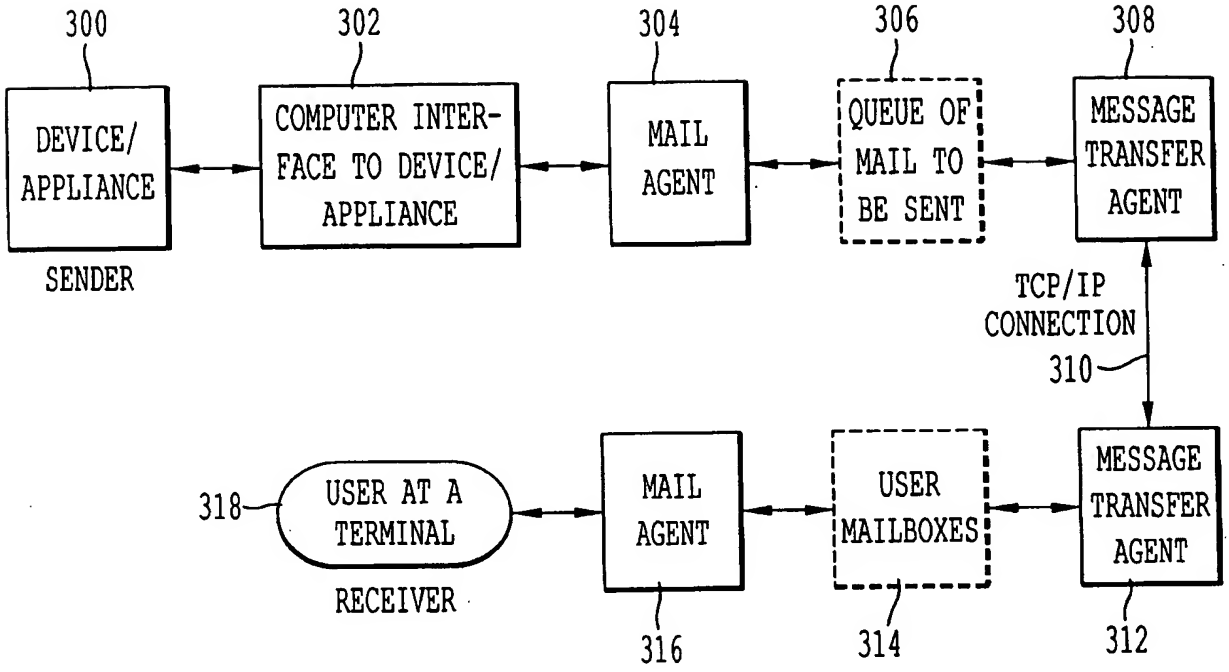
300
302
304
306
308
DEVICE/ APPLIANCE
COMPUTER INTER-FACE TO DEVICE/ APPLIANCE
MAIL AGENT
QUEUE OF MAIL TO BE SENT
MESSAGE TRANSFER AGENT
SENDER
TCP/IP CONNECTION
310
318
USER AT A TERMINAL
MAIL AGENT
USER MAILBOXES
MESSAGE TRANSFER AGENT
RECEIVER
316
314
312

FIG. 6A

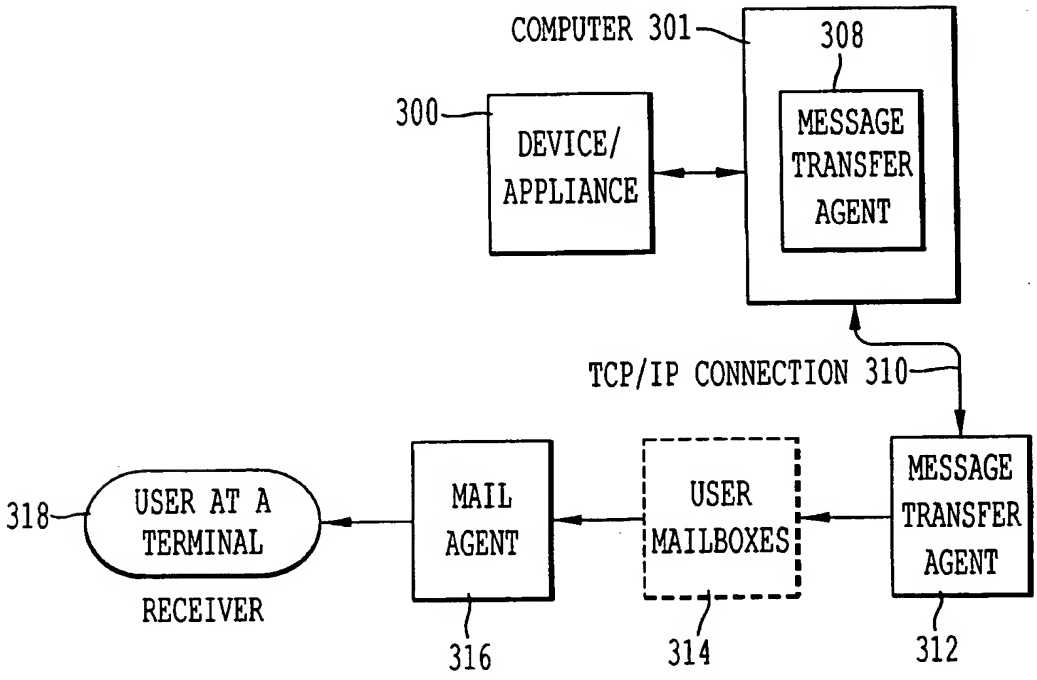
COMPUTER 301
308
300
DEVICE/ APPLIANCE
MESSAGE TRANSFER AGENT
TCP/IP CONNECTION 310
318
USER AT A TERMINAL
MAIL AGENT
USER MAILBOXES
MESSAGE TRANSFER AGENT
RECEIVER
316
314
312

FIG. 6B

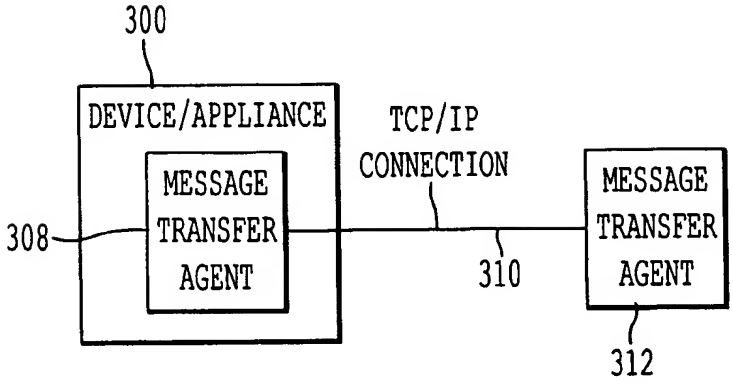
300
DEVICE/APPLIANCE
MESSAGE TRANSFER AGENT
308
TCP/IP CONNECTION
310
MESSAGE TRANSFER AGENT
312

FIG. 6C

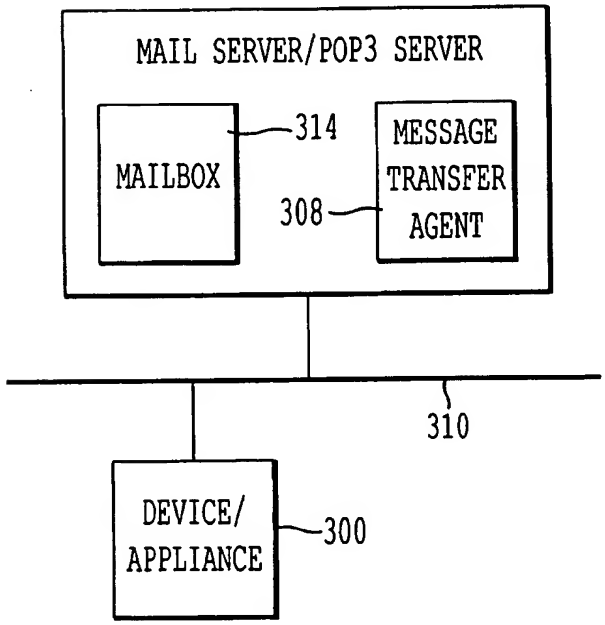
MAIL SERVER/POP3 SERVER
MAILBOX
314
MESSAGE TRANSFER AGENT
308
310
DEVICE/ APPLIANCE
300

FIG. 6D

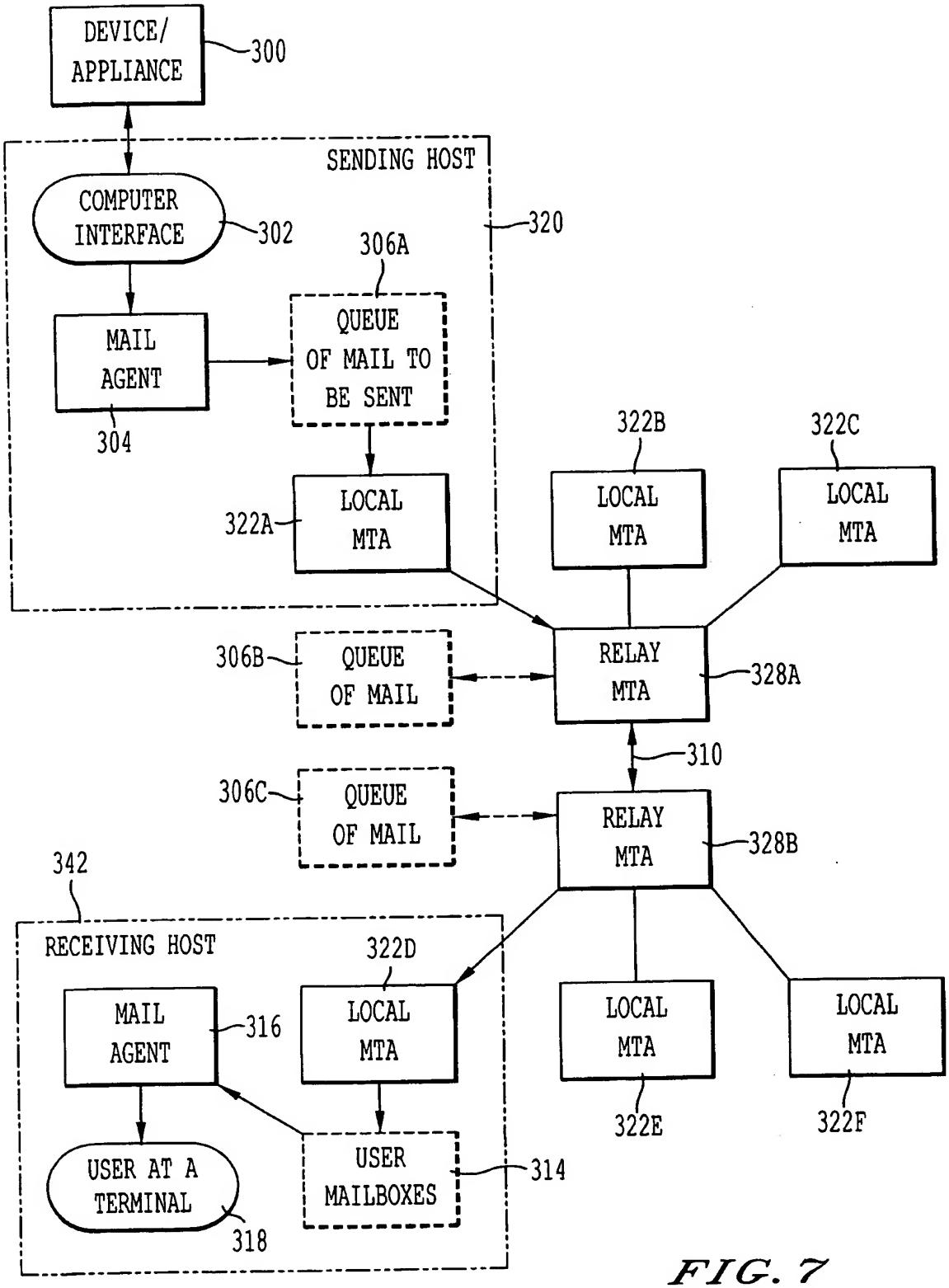
DEVICE/ APPLIANCE
300
SENDING HOST
320
COMPUTER INTERFACE
302
306A
QUEUE OF MAIL TO BE SENT
MAIL AGENT
304
322A
LOCAL MTA
322B
LOCAL MTA
322C
LOCAL MTA
306B
QUEUE OF MAIL
RELAY MTA
328A
310
306C
QUEUE OF MAIL
RELAY MTA
328B
342
RECEIVING HOST
322D
MAIL AGENT
316
LOCAL MTA
LOCAL MTA
322E
LOCAL MTA
322F
USER AT A TERMINAL
318
USER MAILBOXES
314

FIG. 7

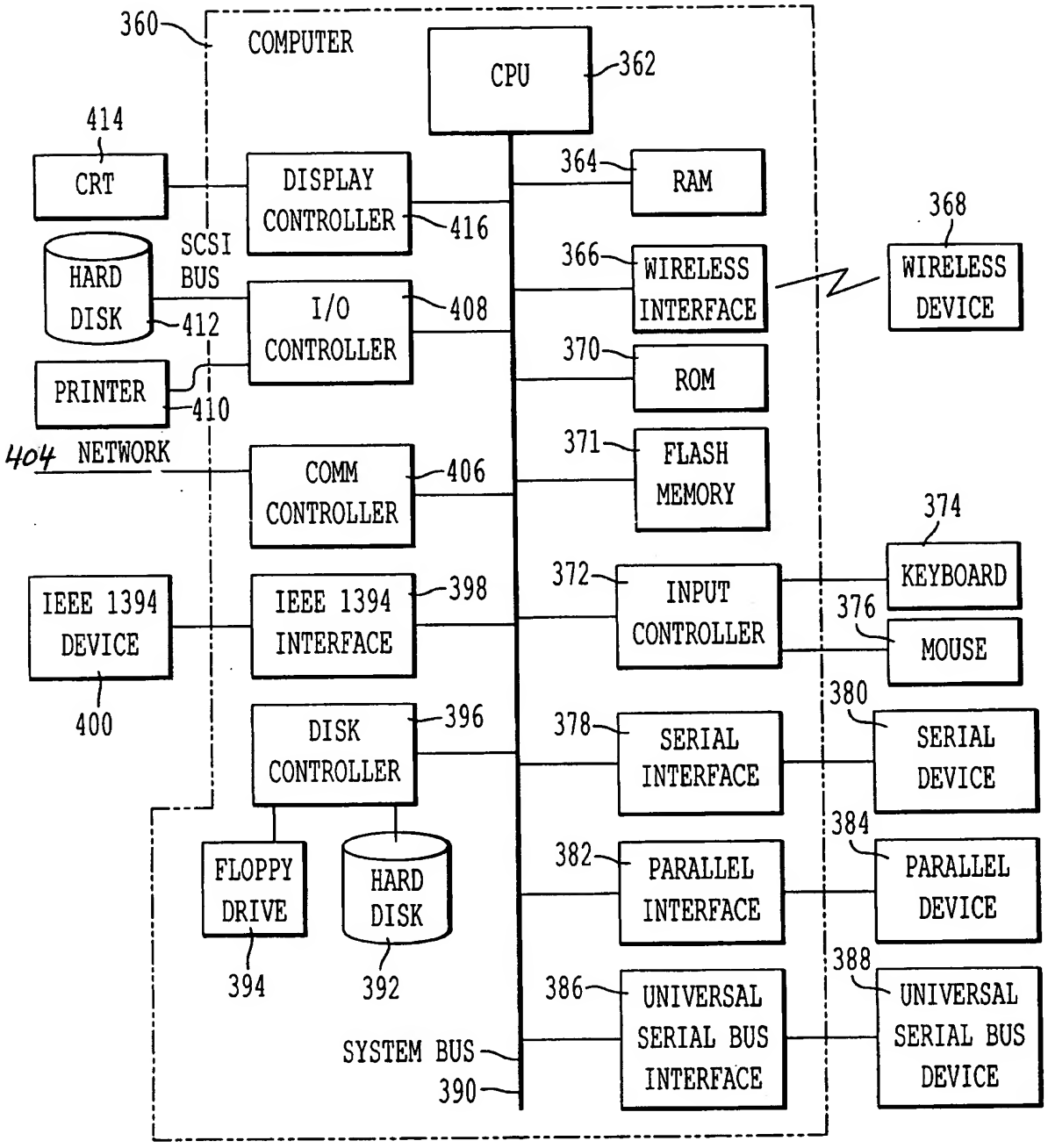
360
COMPUTER
CPU
362
414
CRT
DISPLAY CONTROLLER
416
364
RAM
368
WIRELESS DEVICE
366
WIRELESS INTERFACE
SCSI BUS
HARD DISK
412
I/O CONTROLLER
408
370
ROM
PRINTER
410
404 NETWORK
COMM CONTROLLER
406
371
FLASH MEMORY
374
KEYBOARD
372
INPUT CONTROLLER
376
MOUSE
IEEE 1394 DEVICE
400
IEEE 1394 INTERFACE
398
DISK CONTROLLER
396
380
SERIAL DEVICE
378
SERIAL INTERFACE
384
PARALLEL DEVICE
382
PARALLEL INTERFACE
FLOPPY DRIVE
394
HARD DISK
392
388
UNIVERSAL SERIAL BUS DEVICE
386
UNIVERSAL SERIAL BUS INTERFACE
SYSTEM BUS
390

FIG. 8

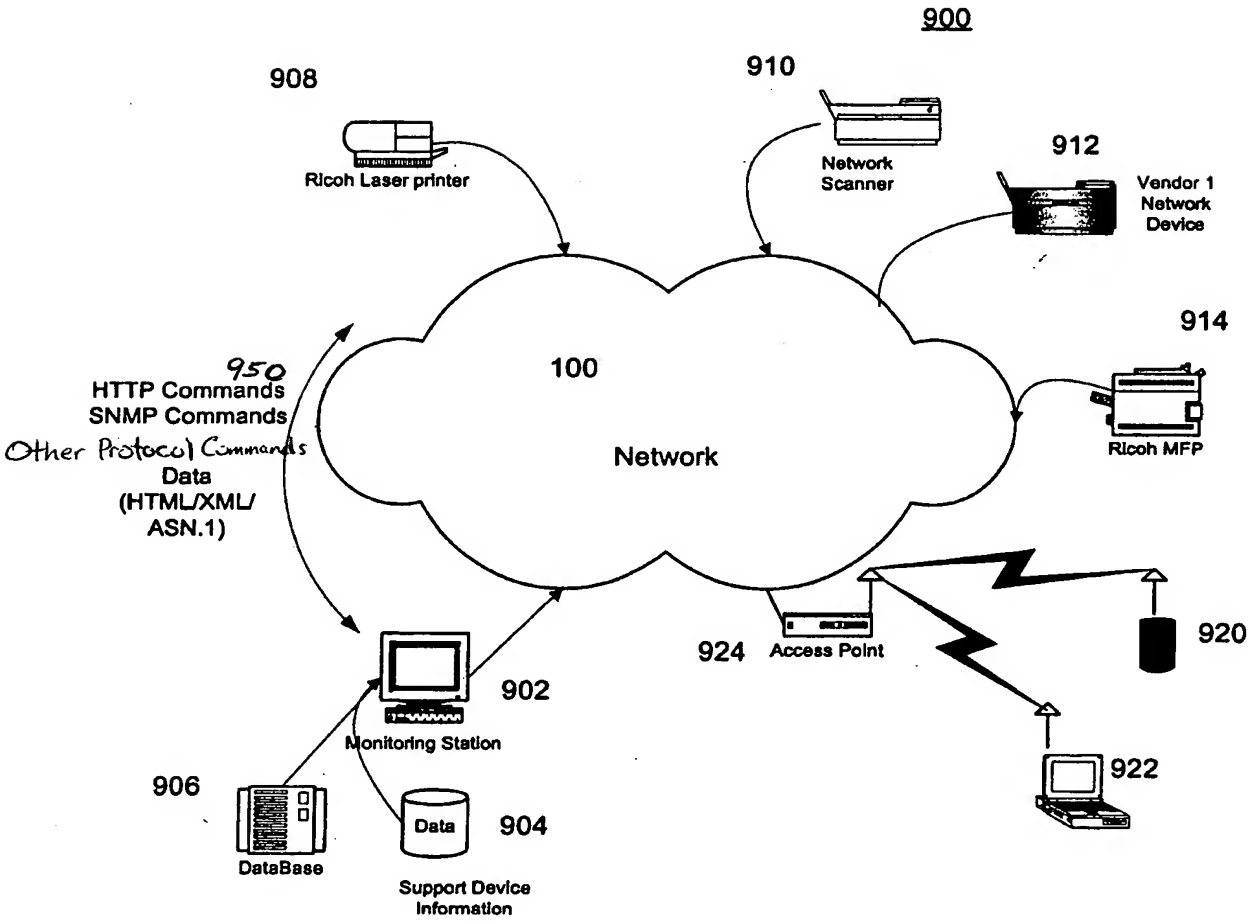
900
908
Ricoh Laser printer
910
Network Scanner
912
Vendor 1 Network Device
914
Ricoh MFP
100
Network
950
HTTP Commands
SNMP Commands
Other Protocol Commands
Data (HTML/XML/ ASN.1)
902
Monitoring Station
906
DataBase
Data
904
Support Device Information
924
Access Point
920
922

Figure 9

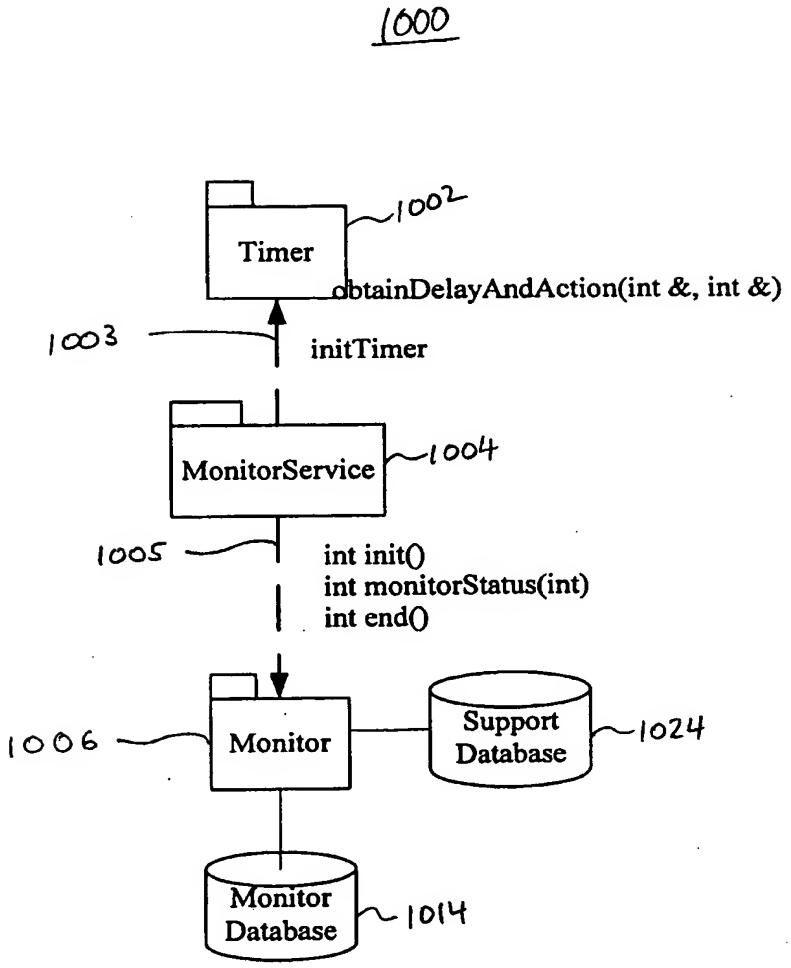
1000
Timer
1002
obtainDelayAndAction(int &, int &)
1003
initTimer
MonitorService
1004
1005
int init()
int monitorStatus(int)
int end()
1006
Monitor
Support Database
1024
Monitor Database
1014

FIG. 10

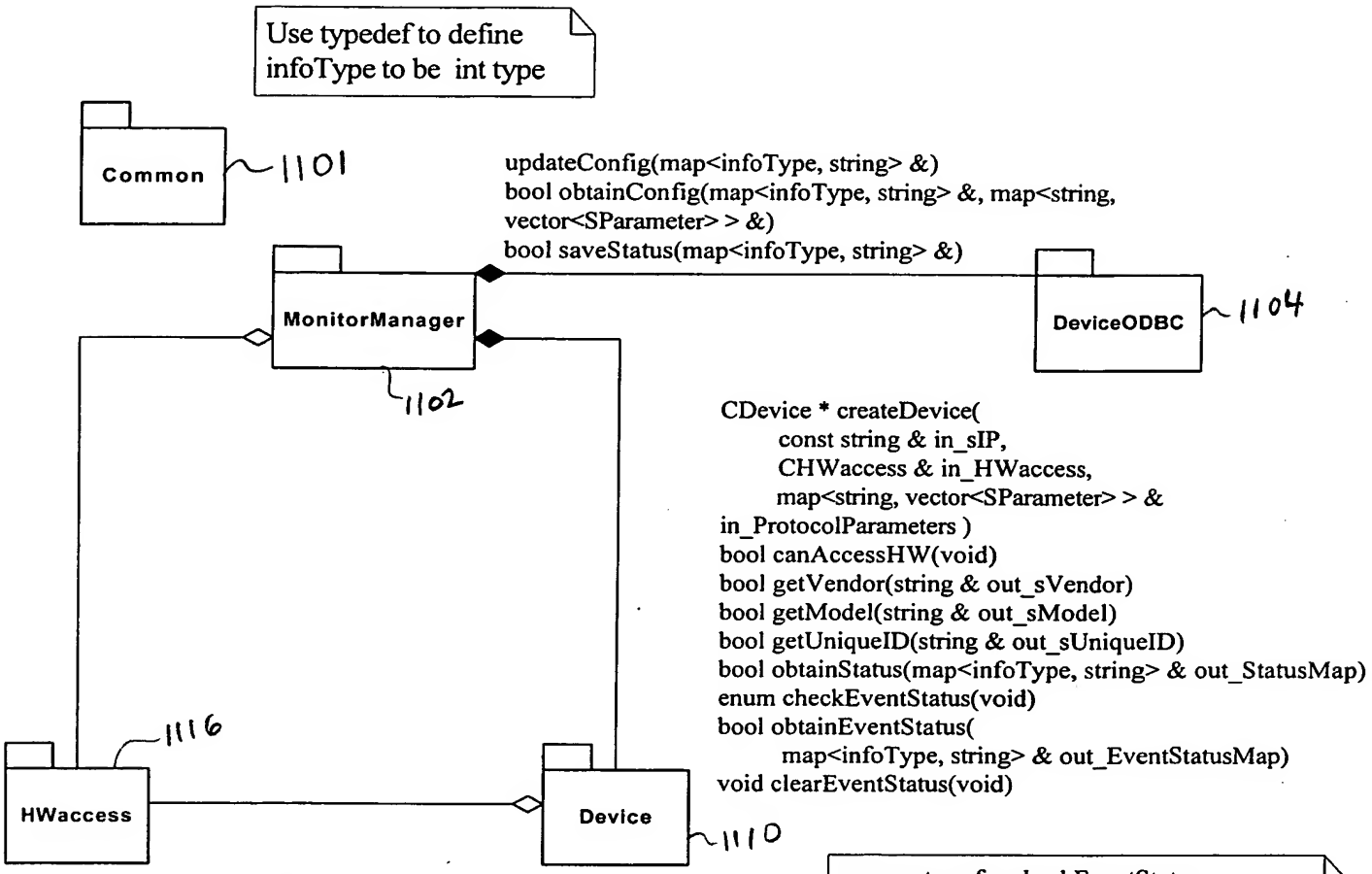
Use typedef to define infoType to be int type
Common
1101
updateConfig(map<infoType, string> &)
bool obtainConfig(map<infoType, string> &, map<string, vector<SParameter> > &)
bool saveStatus(map<infoType, string> &)
MonitorManager
1102
DeviceODBC
1104
CDevice * createDevice(
const string & in_sIP,
CHWaccess & in_HWaccess,
map<string, vector<SParameter> > &
in_ProtocolParameters)
bool canAccessHW(void)
bool getVendor(string & out_sVendor)
bool getModel(string & out_sModel)
bool getUniqueID(string & out_sUniqueID)
bool obtainStatus(map<infoType, string> & out_StatusMap)
enum checkEventStatus(void)
bool obtainEventStatus(
map<infoType, string> & out_EventStatusMap)
void clearEventStatus(void)
1116
HWaccess
Device
1110
void initBegin(void)
void initEnd(void)
bool canAccessIP(
const string & in_sIP,
map<string, vector<SParameter> > & in_ProtocolParameters)
bool obtainVendor(string & out_sVendor,
map<string, vector<SParameter> > & inOut_ProtocolParameters,
const string & in_sIP)
bool obtainModel(string & out_sModelName,
map<string, vector<SParameter> > & inOut_ProtocolParameters,
const string & in_sIP)
bool obtainUniqueID(string & out_sUniqueID,
map<string, vector<SParameter> > & inOut_ProtocolParameters,
const string & in_sIP)
EerrorCode obtainEventStatus(
map<infoType,string> & out_StatusMap,
const string & in_sIP,
map<string, vector<SParameter> > & in_ProtocolParameters)
bool obtainStatus(map<infoType,string> & out_StatusMap,
const string & in_sIP, const string & in_sVendor,
const string & in_sModel,
map<string, vector<SParameter> > & in_ProtocolParameters)
enum return for checkEventStatus should have values eNoEventSinceClearAndNoEventDetected, eNoEventSinceClearAndEventDetected, eEventSinceClearAndNoEventDetected, eEventSinceClearAndEventDetected.

FIG. 11

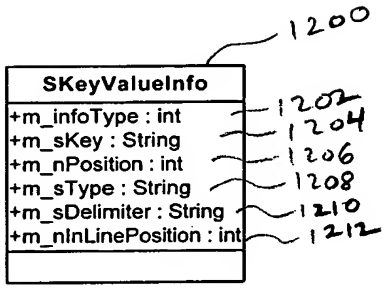
1200
SKeyValueInfo
+m_infoType : int
+m_sKey : String
+m_nPosition : int
+m_sType : String
+m_sDelimiter : String
+m_nInLinePosition : int
1202
1204
1206
1208
1210
1212

FIG. 12

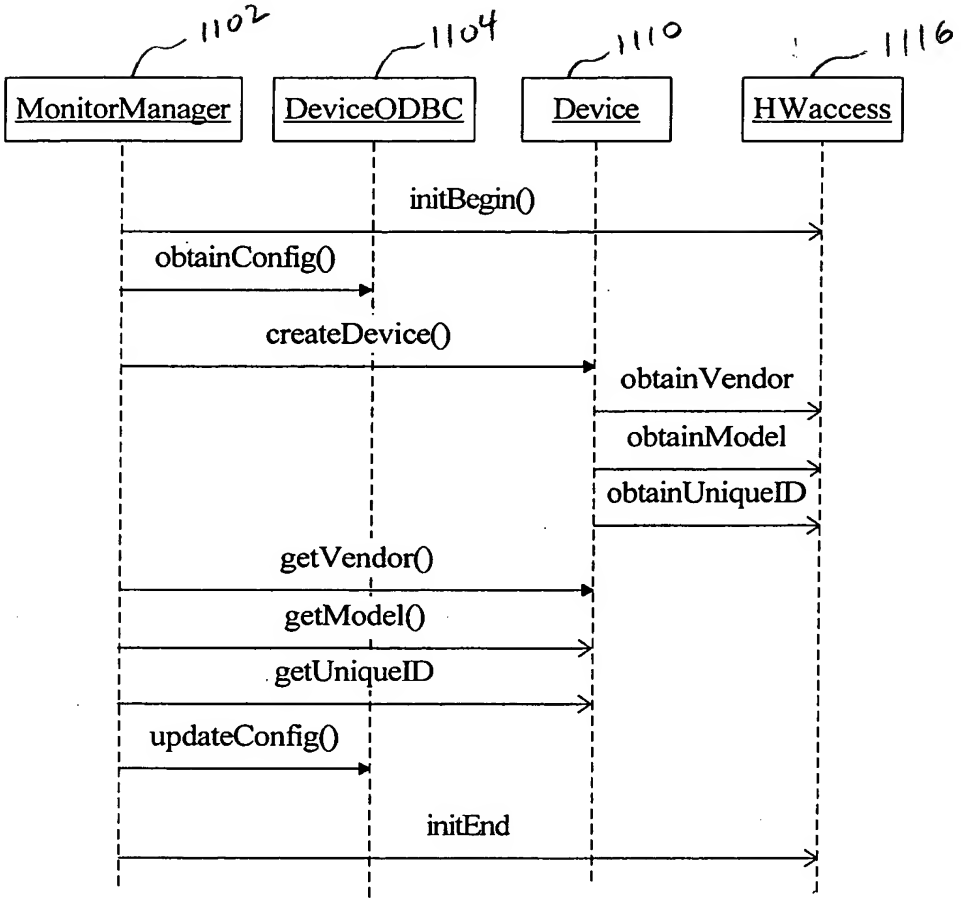
1102
1104
1110
1116
MonitorManager
DeviceODBC
Device
HWaccess
initBegin()
obtainConfig()
createDevice()
obtainVendor
obtainModel
obtainUniqueID
getVendor()
getModel()
getUniqueID
updateConfig()
initEnd

FIG. 13

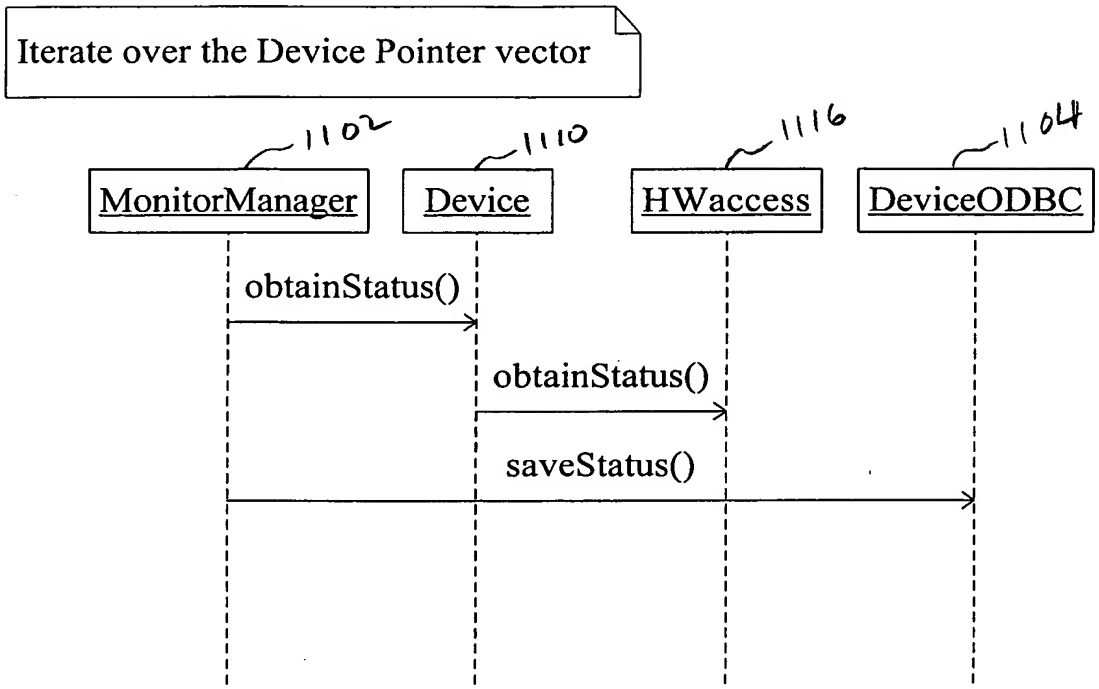
Iterate over the Device Pointer vector
1102
1110
1116
1104
MonitorManager
Device
HWaccess
DeviceODBC
obtainStatus()
obtainStatus()
saveStatus()

FIG. 14

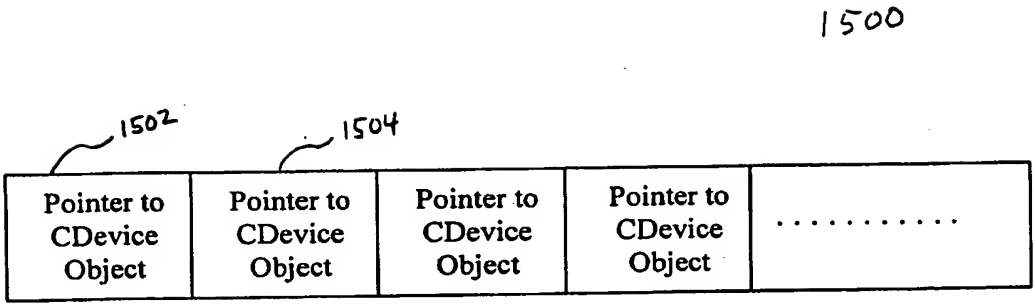
1500
1502
1504
Pointer to CDevice Object
Pointer to CDevice Object
Pointer to CDevice Object
Pointer to CDevice Object
.....
std::vector <pointer of CDevice >

FIG. 15

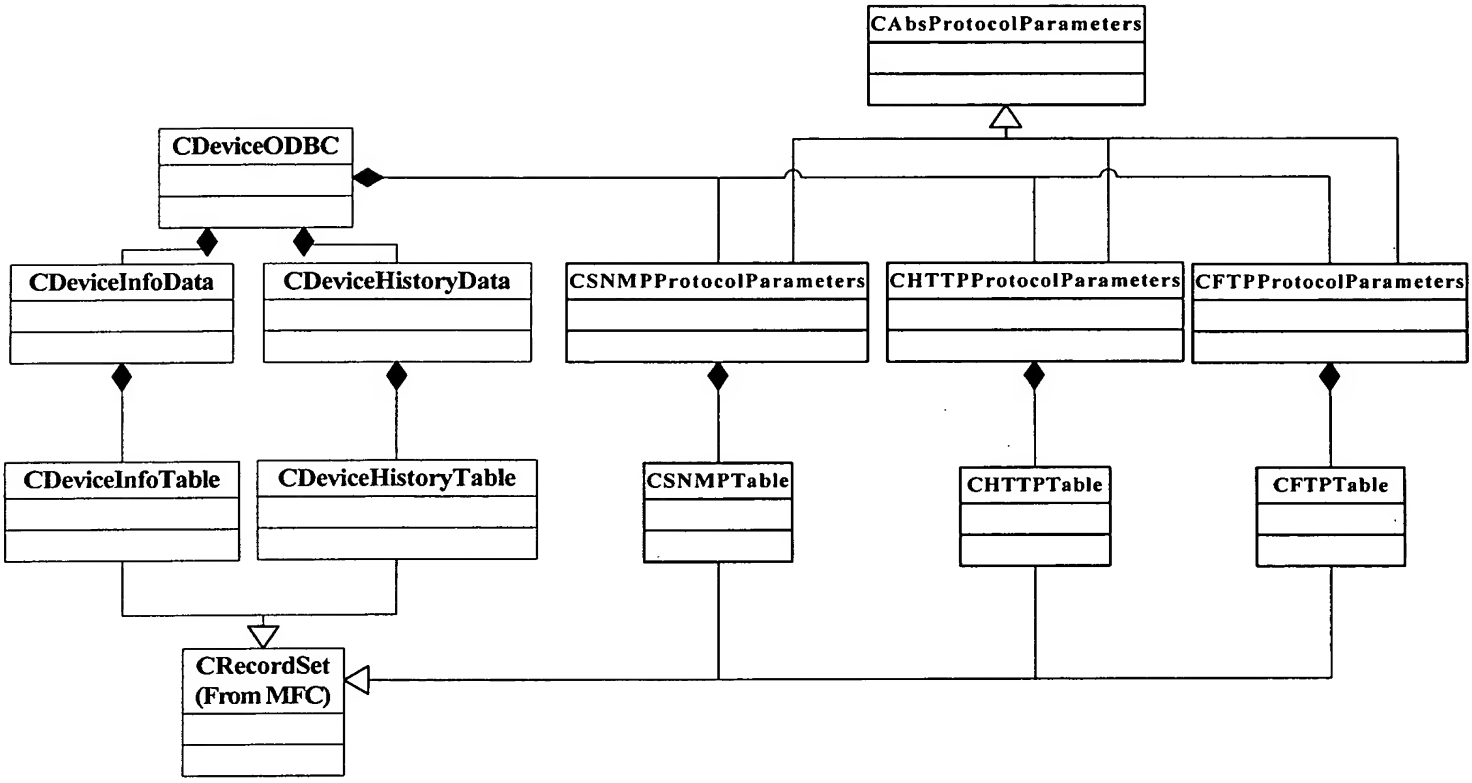
CAbsProtocolParameters
CDeviceODBC
CDeviceInfoData
CDeviceHistoryData
CSNMPProtocolParameters
CHTTPProtocolParameters
CFTPProtocolParameters
CDeviceInfoTable
CDeviceHistoryTable
CSNMPTable
CHTTPTable
CFTPTable
CRecordSet
(From MFC)

FIG. 16

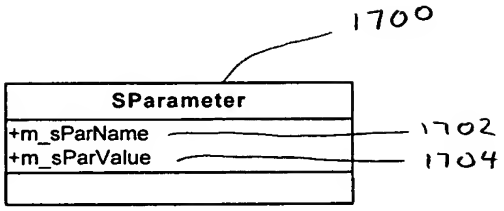
1700
SParameter
+m_sParName
+m_sParValue
1702
1704

FIG. 17

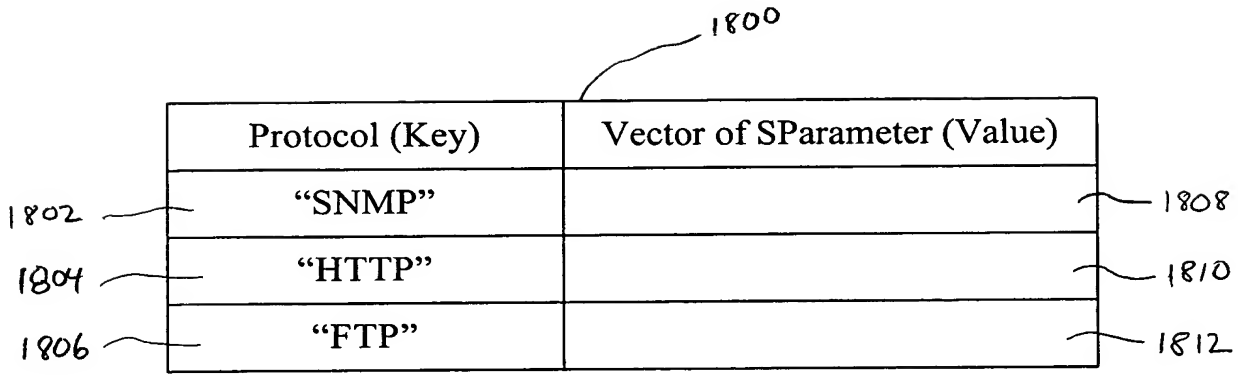
1800
Protocol (Key)
Vector of SParameter (Value)
1802
"SNMP"
1808
1804
"HTTP"
1810
1806
"FTP"
1812

FIG. 18

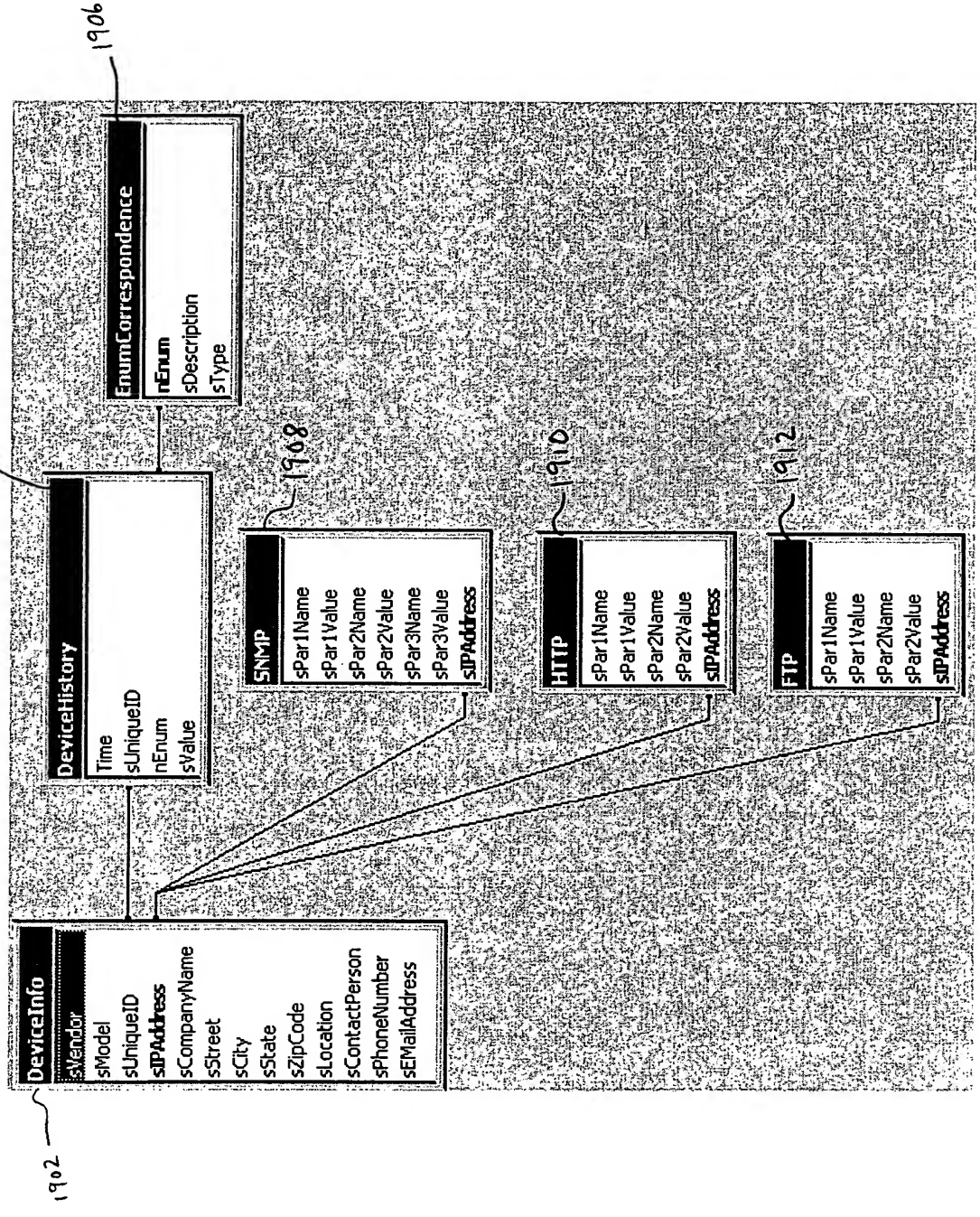
1902
DeviceInfo
sVendor
sModel
sUniqueID
sIPAddress
sCompanyName
sStreet
sCity
sState
sZipCode
sLocation
sContactPerson
sPhoneNumber
sEMailAddress
1904
DeviceHistory
Time
sUniqueID
nEnum
sValue
1906
EnumCorrespondence
nEnum
sDescription
sType
1908
SNMP
sPar1Name
sPar1Value
sPar2Name
sPar2Value
sPar3Name
sPar3Value
sIPAddress
1910
HTTP
sPar1Name
sPar1Value
sPar2Name
sPar2Value
sIPAddress
1912
FTP
sPar1Name
sPar1Value
sPar2Name
sPar2Value
sIPAddress

FIG. 19

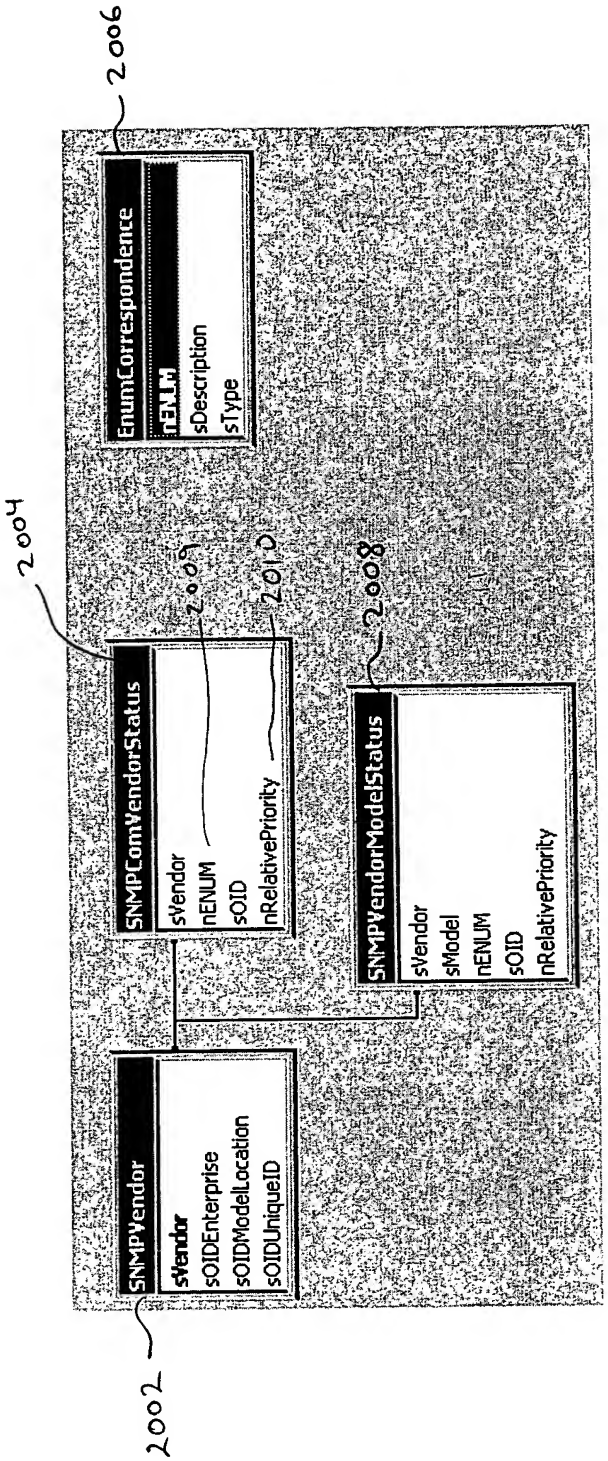
2006
EnumCorrespondence
nENUM
sDescription
sType
2004
SNMPComVendorStatus
sVendor
nENUM
sOID
nRelativePriority
2009
2010
2008
SNMPVendorModelStatus
sVendor
sModel
nENUM
sOID
nRelativePriority
SNMPVendor
sVendor
sOIDEnterprise
sOIDModelLocation
sOIDUniqueID
2002

FIG. 20

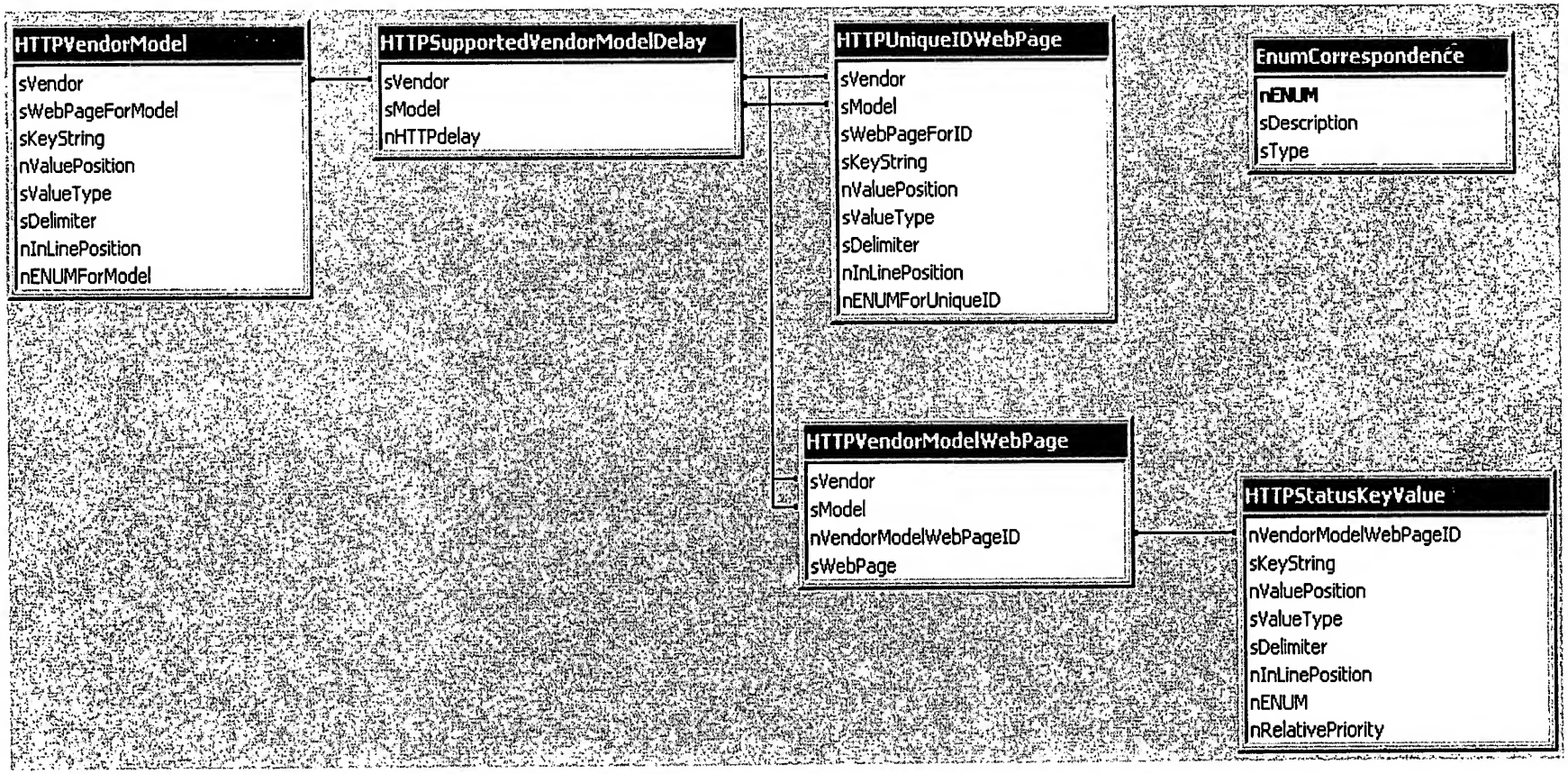

FIG. 21

OBLON, SPIVAK, ET AL
DOCKET #: 240204US-28
INV: Tetsuro MOTOYAMA
SHEET 21 OF 25

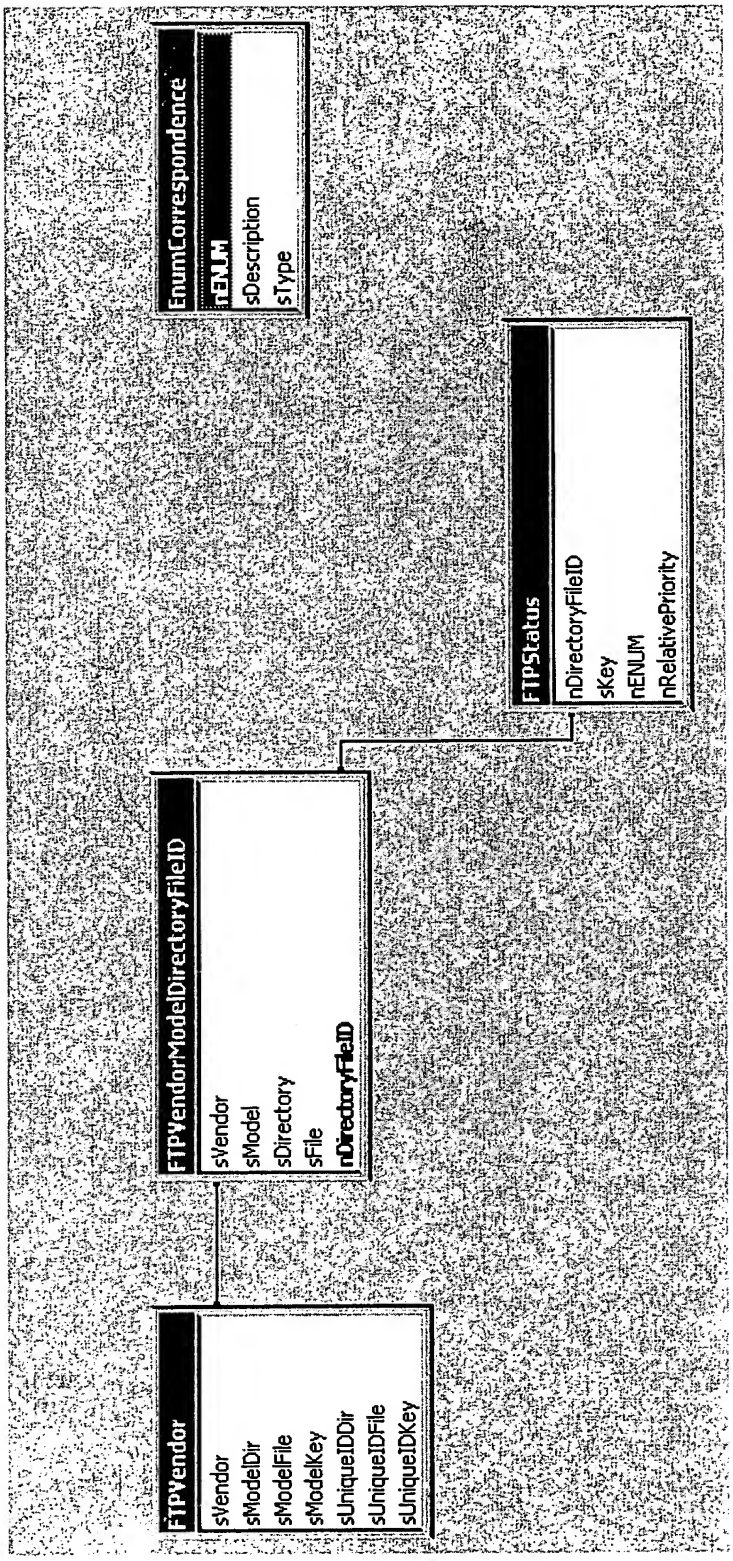
FTPVendor
sVendor
sModelDir
sModelFile
sModelKey
sUniqueIDDir
sUniqueIDFile
sUniqueIDKey
FTPVendorModelDirectoryFileID
sVendor
sModel
sDirectory
sFile
nDirectoryFileID
EnumCorrespondence
nENUM
sDescription
sType
FTPStatus
nDirectoryFileID
sKey
nENUM
nRelativePriority

FIG. 22

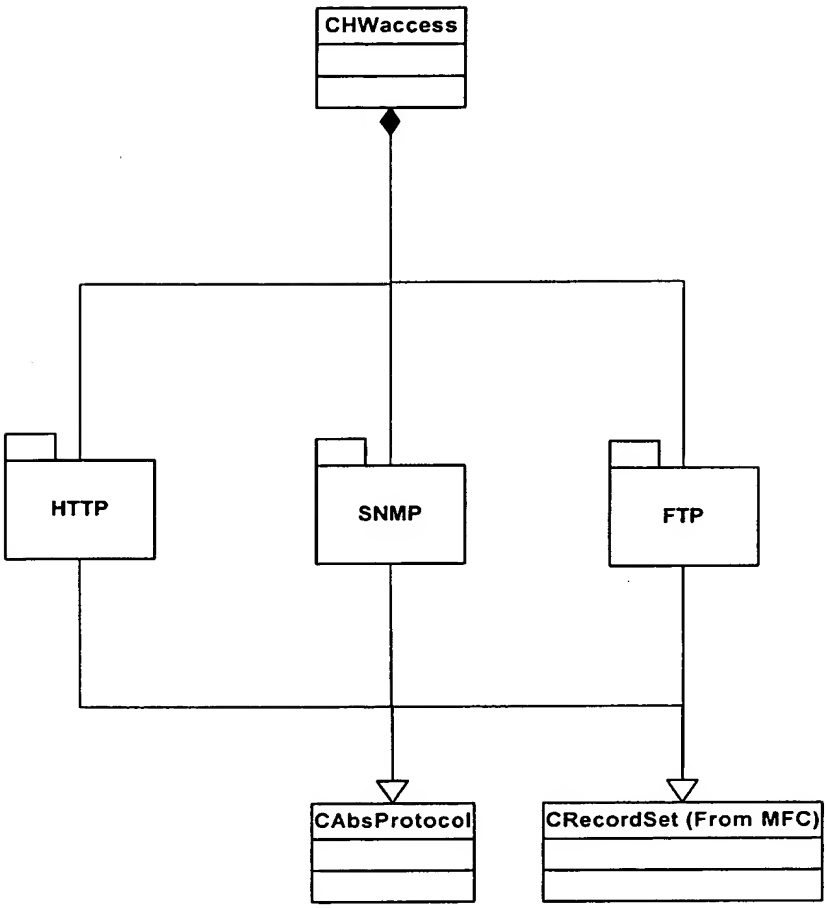

FIG. 23

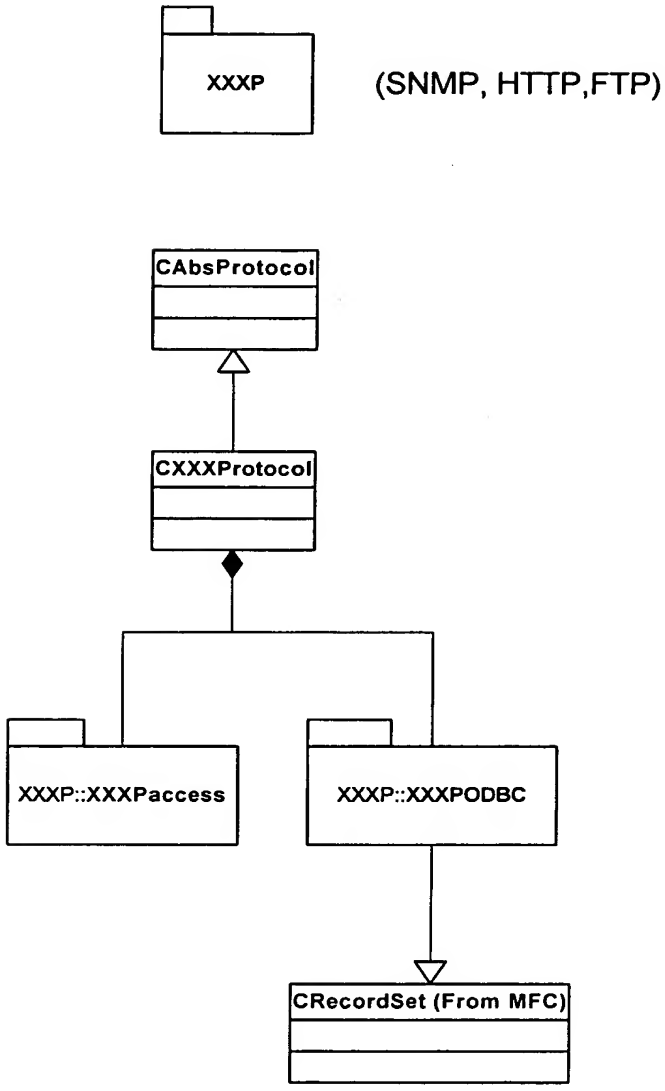
XXXP
(SNMP, HTTP,FTP)
CAbsProtocol
CXXXProtocol
XXXP::XXXPaccess
XXXP::XXXPODBC
CRecordSet (From MFC)

FIG. 24

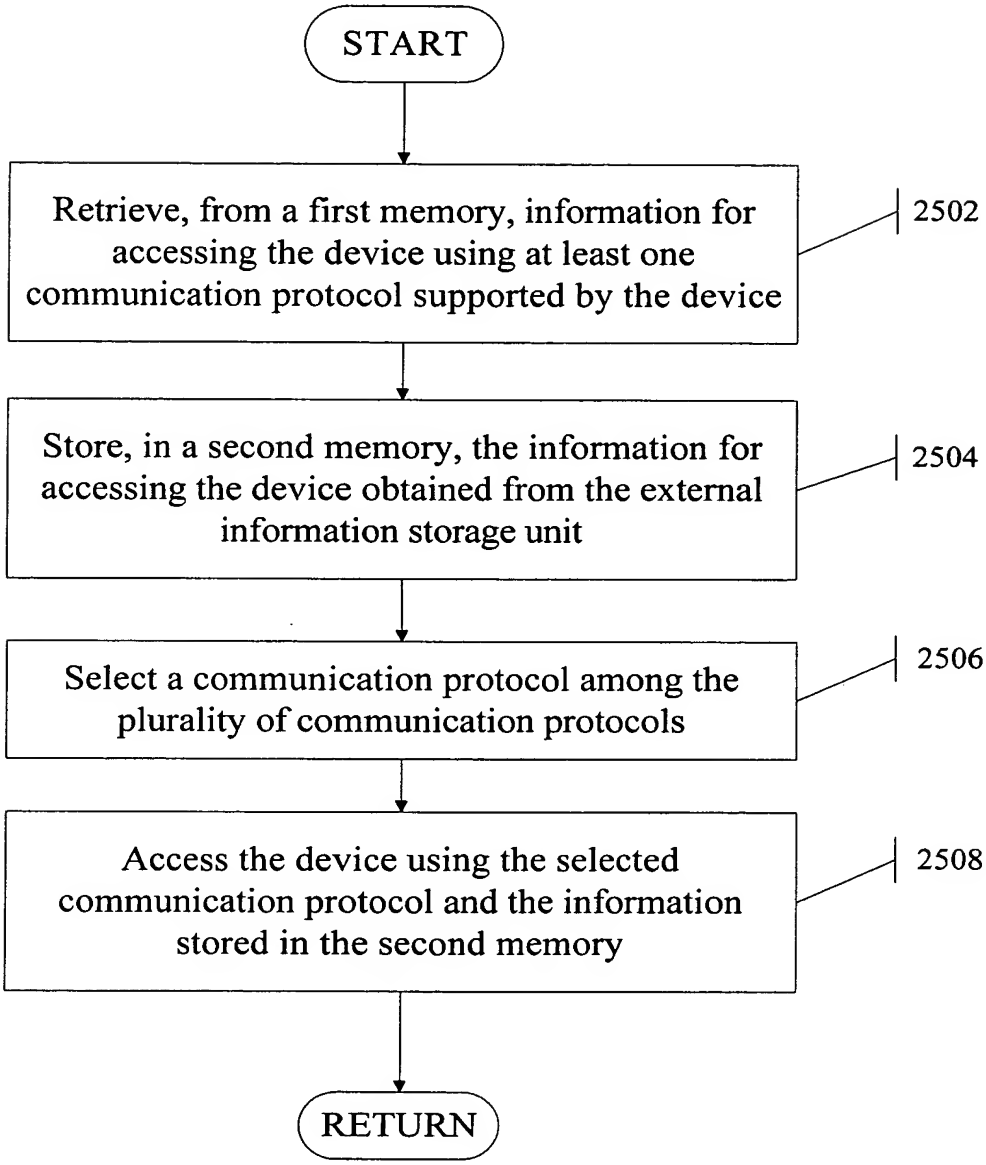
START
Retrieve, from a first memory, information for accessing the device using at least one communication protocol supported by the device
2502
Store, in a second memory, the information for accessing the device obtained from the external information storage unit
2504
Select a communication protocol among the plurality of communication protocols
2506
Access the device using the selected communication protocol and the information stored in the second memory
2508
RETURN

FIG. 25